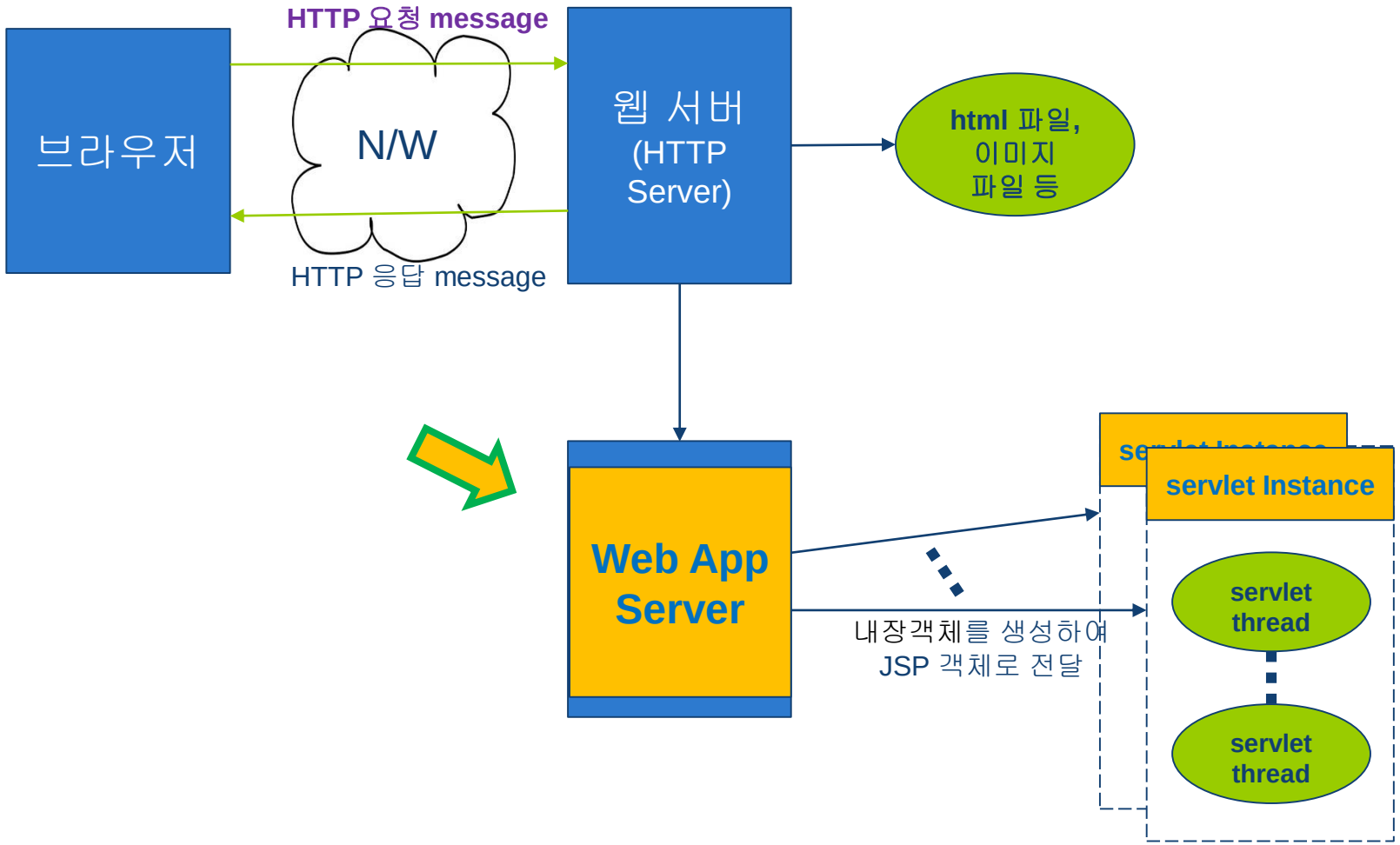
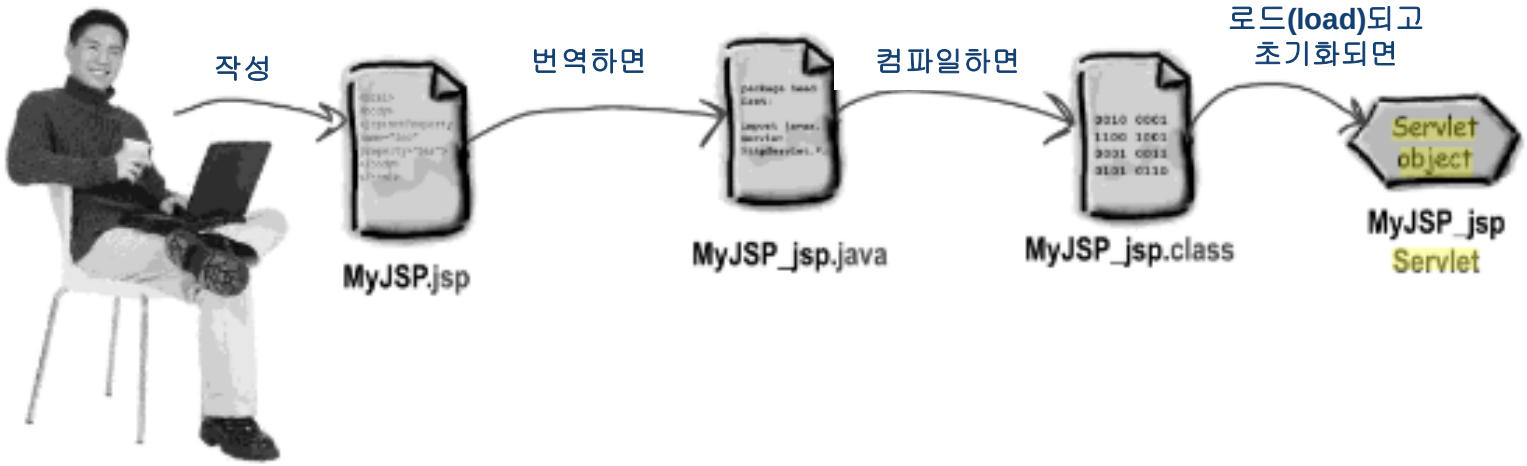
제 06 장
JSP 액션 태그

2020년도 1학기

웹어플리케이션 다시!


HTTP 요청 message
브라우저
N/W
웹 서버
(HTTP
Server)
html 파일,
이미지
파일 등
HTTP 응답 message
Web App
Server
servlet Instance
servlet
thread
servlet
thread
내장객체를 생성하여
JSP 객체로 전달

JSP 파일의 변천


작성
MyJSP.jsp
번역하면
MyJSP_jsp.java
컴파일하면
MyJSP_jsp.class
로드(load)되고
초기화되면
Servlet
object
MyJSP_jsp
Servlet

초심자 플린의 JSP 처리 시도

BasicCounter.jsp

```
<html>
<body>
The page count is:
<%
  out.println(Counter.getCount());
%>
</body>
</html>
```

The "out" object is implicitly there. Everything between <% and %> is a scriptlet, which is just plain old Java.

Counter.java

```
package foo;

public class Counter {
   private static int count;
   public static synchronized int getCount()    {
      count++;
      return count;
   }
}
```

Plain old Java helper class.

뭐가 잘못된 거죠?

What she expected:

The page count is: 1

What she got:

HTTP Status 500 -

The server encountered an internal error () that prevented it from fulfilling this request.

exception org.apache.jasper.JasperException: Unable to compile class for JSP

An error occurred at line: 1 in the jsp file: /BasicCounter.jsp
Generated servlet error:
[javac] Compiling 1 source file
/Users/kathy/Applications2/jakarta-tomcat-5.0.19/work/Catalina/localhost/testJSP1/org/apache/jsp/BasicCounter_jsp.java:45: cannot resolve symbol
symbol : variable Counter
location: class org.apache.jsp.basicCounter_jsp
out.print(Counter.getCount());
^

Can you figure out what's wrong?

1 error

```
org.apache.jasper.compiler.DefaultErrorHandler.javacError(DefaultErrorHandler.java:127)
org.apache.jasper.compiler.ErrorDispatcher.javacError(ErrorDispatcher.java:351)
org.apache.jasper.compiler.Compiler.generateClass(Compiler.java:415)
org.apache.jasper.compiler.Compiler.compile(Compiler.java:458)
org.apache.jasper.compiler.Compiler.compile(Compiler.java:439)
org.apache.jasper.JspCompilationContext.compile(JspCompilationContext.java:553)
org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:291)
org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:301)
org.apache.jasper.servlet.JspServlet.service(JspServlet.java:248)
javax.servlet.http.HttpServlet.service(HttpServlet.java:856)
```

import 실수

Counter.java

```
package foo;

public class Counter {
   private static int count;
   public static int getCount() {
       count++;
       return count;
   }
}
```

JSP code was:

```
<% out.println(Counter.getCount()); %>
```

JSP code *should be*:

```
<% out.println(foo.Counter.getCount()); %>
```

Now it'll work.

page 지시자로 import 하기

To import a *single* package:

```
<%@ page import="foo.*" %>

<html>
<body>
The page count is:
<%
  out.println(Counter.getCount());
%>
</body>
</html>
```

This is a page directive with an import attribute.

(Notice there's no semicolon at the end of a directive.)

Scriptlets are normal Java, so all statements in a scriptlet must end in a semicolon!

To import *multiple* packages:

```
<%@ page import="foo.*,java.util.*" %>
```

Use a comma to separate the packages.
The quotes go around the entire list of packages!

print() vs Expression

Scriptlet code:

```
<%@ page import="foo.*" %>
<html>
<body>
The page count is:
<% out.println(Counter.getCount()); %>
</body>
</html>
```

Expression code:

```
<%@ page import="foo.*" %>
<html>
<body>
The page count is now:
<%= Counter.getCount() %>
</body>
</html>
```

The expression is shorter—we don't need to explicitly do the print...

Expression이 승리!

Expressions become the argument to an out.print()

In other words, the Container takes *everything* you type between the <%= and %> and puts it in as the argument to a statement that prints to the implicit response PrintWriter *out*.

When the Container sees *this:*

```
<%= Counter.getCount() %>
```

It turns it into *this:*

```
out.print(Counter.getCount());
```

If you *did* put a semicolon in your expression:

```
<%= Counter.getCount(); %>
```

That would be bad. It would mean *this:*

```
out.print(Counter.getCount(););
```

Yikes!! This will never compile.

다르게 해결하기

What she tried:

```
<html>
<body>
<% int count=0; %>
The page count is now:
<%= ++count %>
</body>
</html>
```

Will it *compile*?

Will it *work*?

스크립트릿으로 처리

What she tried:

```
<html>
<body>
<% int count=0; %>
The page count is now:
<%= ++count %>
</body>
</html>
```

We don't need to import anything, so we dropped the page directive.

scriptlet

Declare the count variable.

expression

Increment the count variable and print the value.

What she got the first time she hit the page:

The page count is: 1

Looks good!

What she got the second, third, and every other time she hit the page:

The page count is: 1

Uh-oh... it's still showing "1"

It keeps resetting the variable...

스크립트릿 내 변수의 지위

This JSP:

```
<html><body>
<% int count=0; %>
The page count is now:
<%= ++count %>
</body></html>
```

Becomes this servlet:

```
public class basicCounter_jsp extends SomeSpecialHttpServlet {

  public void _jspService(HttpServletRequest request,
      HttpServletResponse response)throws java.io.IOException,
                                          ServletException {

    PrintWriter out = response.getWriter();
    response.setContentType("text/html");
    out.write("<html><body>");
    int count=0;
    out.write("The page count is now:");
    out.print( ++count );
    out.write("</body></html>");

  }
}
```

The Container puts all the code into a generic service method. Think of it as a catch-all combo doGet/doPost

선언(Declaration)

Variable Declaration

This JSP:

```
<html><body>
<%! int count=0; %>
The page count is now:
<%= ++count %>
</body></html>
```

Becomes this servlet:

```
public class basicCounter_jsp extends HttpServlet {

    int count=0;

    public void _jspService(HttpServletRequest request,
      HttpServletResponse response)throws java.io.IOException {

        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        out.write("<html><body>");
        out.write("The page count is now:");
        out.print( ++count );
        out.write("</body></html>");
    }
}
```

This time, we're incrementing an instance variable instead of a local variable.

Method Declaration

This JSP:

```
<html>
<body>
<%! int doubleCount() {
      count = count*2;
      return count;
    }
%>
<%! int count=1; %>
The page count is now:
<%= doubleCount() %>
</body>
</html>
```

Becomes this servlet:

```
public class basicCounter_jsp extends HttpServlet {

    int doubleCount() {
        count = count*2;
        return count;
    }
    int count=1;

    public void _jspService(HttpServletRequest request,
      HttpServletResponse response)throws java.io.IOException {
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        out.write("<html><body>");
        out.write("The page count is now:");
        out.print( doubleCount() );
        out.write("</body></html>");
    }
}
```

The method goes in just the way you typed it in your JSP.

It's Java, so no problem with forward-referencing (declaring the variable AFTER you used it in a method).

실제 코드

Tomcat 5 generated class

```
package org.apache.jsp;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
```

If you have page directive imports, they'll show up here (we didn't have any imports for this JSP).

```
<html><body>
<%! int count=0; %>
The page count is now:
<%= ++count %>
</body></html>
```

```
public final class BasicCounter_jsp extends org.apache.jasper.runtime.HttpJspBase
                        implements org.apache.jasper.runtime.JspSourceDependent {

   int count=0;
   private static java.util.Vector _jspx_dependants;

   public java.util.List getDependants() {
     return _jspx_dependants;
   }

   public void _jspService(HttpServletRequest request, HttpServletResponse response)
                                  throws java.io.IOException, ServletException {
      JspFactory _jspxFactory = null;
      PageContext pageContext = null;
      HttpSession session = null;
      ServletContext application = null;
      ServletConfig config = null;
      JspWriter out = null;
      Object page = this;
      JspWriter _jspx_out = null;
      PageContext _jspx_page_context = null;

      try {
        _jspxFactory = JspFactory.getDefaultFactory();
        response.setContentType("text/html");
        pageContext = _jspxFactory.getPageContext(this, request, response,
                  null, true, 8192, true);
        _jspx_page_context = pageContext;
        application = pageContext.getServletContext();
        config = pageContext.getServletConfig();
        session = pageContext.getSession();
        out = pageContext.getOut();
        _jspx_out = out;
        out.write("\r<html>\r<body>\r");
        out.write("\rThe page count is now: \r");
        out.print( ++count );
        out.write("\r</body>\r</html>\r");
      } catch (Throwable t) {
         if (!(t instanceof SkipPageException)){
           out = _jspx_out;
           if (out != null && out.getBufferSize() != 0)
             out.clearBuffer();
           if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
         }
```

The Container puts YOUR declarations (things inside <%! %> tags) and any of its own below the class declaration.

The Container declares a bunch of its own local variables, including those that represent the "implicit objects" your code might need, like "out" and "request".

Now it tries to initialize the implicit objects.

And it tries to run and output your JSP HTML, scriptlet, and expression code.

Of course things might go wrong...

내장객체들

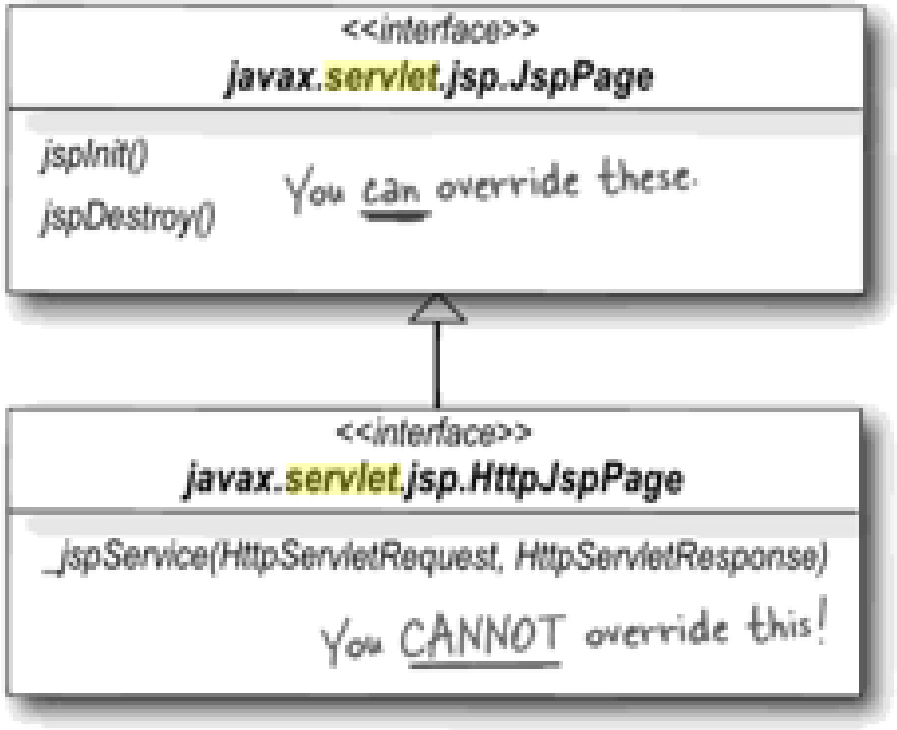
API		Implicit Object
JspWriter	_____	out
HttpServletRequest	_____	request
HttpServletResponse	_____	response
HttpSession	_____	session
ServletContext	_____	application
ServletConfig	_____	config
JspException	_____	exception
PageContext	_____	pageContext
Object	_____	page

Which of these represent the attribute scopes of request, session, and application? (OK, pretty obvious). But now there's a NEW fourth scope, "page-level", and page-scoped attributes are stored in pageContext.

This implicit object is only available to designated "error pages". (You'll see that later in the book.)

A PageContext encapsulates other implicit objects, so if you give some helper object a PageContext reference, the helper can use that reference to get references to the OTHER implicit objects and attributes from all scopes.

상속관계


<<interface>>
javax.servlet.jsp.JspPage
jspInit()
jspDestroy()
You can override these.
<<interface>>
javax.servlet.jsp.HttpJspPage
_jspService(HttpServletRequest, HttpServletResponse)
You CANNOT override this!

액션 태그의 유형

- XML 스타일의 태그로 기술
  - 특정한 동작 기능을 수행
    - <jsp:태그키워드 태그속성="*태그값*" />
    - <jsp:include page="*sub.jsp*" />
- 액션 태그에서 매개변수 지정
  - 시작 태그 <jsp:태그키워드 … >와 종료 태그 </jsp:태그키워드> 사이에 <jsp:param … />과 같은 param 태그를 기술
    - <jsp:태그키워드 태그속성="*태그값*" >
    - 매개변수 지정과 같은 다른 내용
    - </jsp:태그키워드>
    -
    - <jsp:include page="*includesub.jsp*" >
    - <jsp:param name="*weeks*" value="*52*" />
    - </jsp:include>

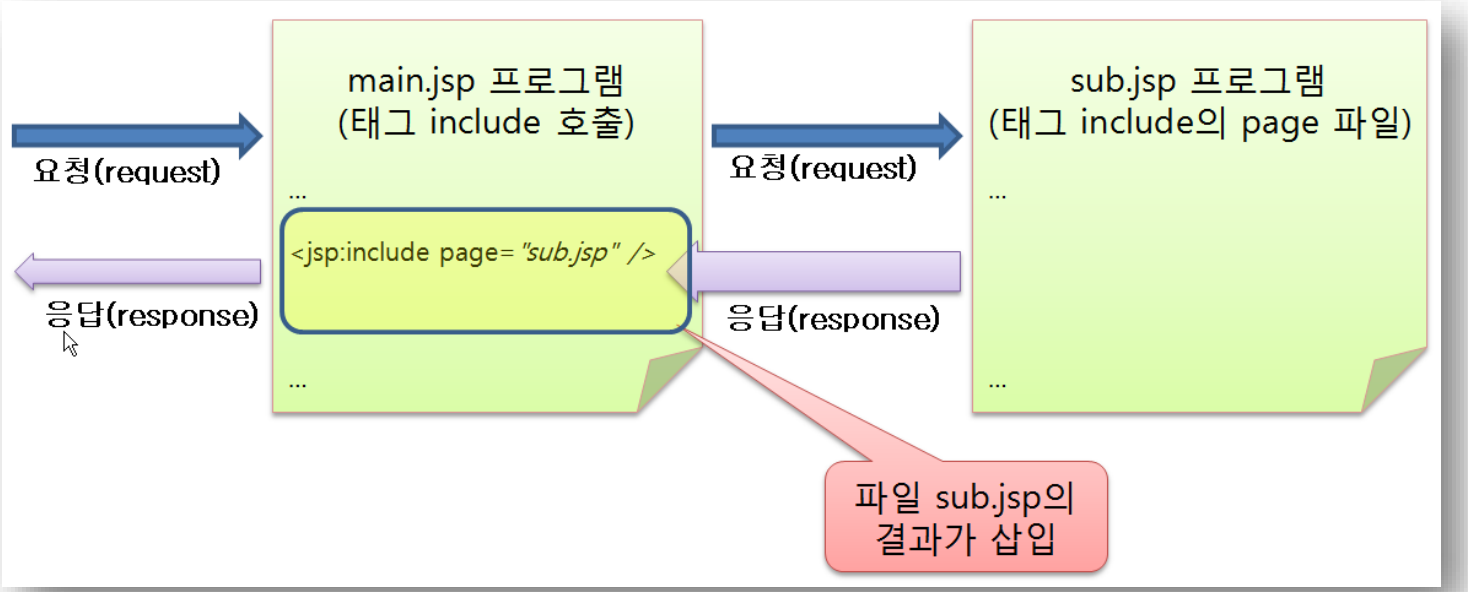
액션 태그 종류

태그 종류	태그 형식	사용 용도
include param	<jsp:include page="test.jsp" /> <jsp:include page="test.jsp" > <jsp:param name="id" value="hong" /> </jsp:include>	현재 JSP 페이지에서 다른 페이지를 포함
forward Param	<jsp:forward page="test.jsp" /> <jsp:forward page="test.jsp" > <jsp:param name="id" value="hong" /> </jsp:forward>	현재 JSP 페이지의 제어를 다른 페이지에 전달
plugin	<jsp:plugin type="applet" code="test" />	자바 애플릿 등을 플러그인
useBean	<jsp:useBean id="login" class="LoginBean" />	자바 빈즈를 사용
setProperty	<jsp:setProperty name="login" property="pass" />	자바 빈즈의 속성을 지정하는 메소드를 호출
getProperty	<jsp:getProperty name="login" property="pass" />	자바 빈즈의 속성을 반환하는 메소드를 호출

액션 태그 include

❖ 속성 page

- 액션 태그 include는 현재의 JSP 페이지에서 기술된 다른 JSP 페이지를 호출하여 그 결과를 include 태그의 위치에 삽입시키는 역할을 수행
- 태그 include에서 속성 page에 삽입할 파일이름을 기술
  - <jsp:include page="*sub.jsp*" />



예제

❖ main.jsp, sub.jsp

main.jsp 프로그램

```
<h2> include 액션 태그 </h2>
main.jsp 파일 시작 부분입니다.<br>
include 태그는 페이지 속성 파일 결과를 태그 위치에 삽입합니다.<br>

<jsp:include page="sub.jsp" />

main.jsp 파일 끝 부분입니다.
```

JSP 예제 : sub.jsp

http://localhost:8080/ch06/sub.jsp

이 부분은 include 태그가 있던 자리로 sub.jsp의 결과가 삽입됩니다.

파일 sub.jsp의 결과가 삽입

JSP 예제 : main.jsp

http://localhost:8080/ch06/main.jsp

include 액션 태그

main.jsp 파일 시작 부분입니다.
include 태그는 페이지 속성 파일 결과를 태그 위치에 삽입합니다.

이 부분은 include 태그가 있던 자리로 sub.jsp의 결과가 삽입됩니다.

main.jsp 파일 끝 부분입니다.

지시자 include

- 소스의 삽입
- 변수의 선언이 중복될 경우, 오류가 발생!
  - 지시자 include가 있는 페이지 includedirective.jsp에 변수 i와 n이 선언되었다고 가정
    - <% int i = 12; %>
    - <% int n = 365; %>
    - <%@ include file="includesub.jsp" %>
  - 소스가 삽입되는 페이지 includesub.jsp
    - <% int n = 52; %>

includedirective.jsp

```
1 <%@ page language="java" contentType="text/htm
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/
5 <title>JSP 예제 : includedirective.jsp</title>
6 </head>
7 <body>
8     <% int i = 12; %>
9     <% int n = 365; %>
10    <% int days = 365; %>
11    1 년은 <%=i %>달 입니다.
12    Duplicate local variable n  e file="includesub.jsp" %>
13    1 년은 <%=days %>일 입니다.
14 </body>
15 </html>
```

액션 태그 include

❖ 결과의 삽입

- 지시자 include와 다르게 액션 태그 include를 이용했을 경우는 결과값이 포함되기 때문에 이러한 지역변수 중복 선언의 문제가 발생하지 않음

❖ 액션 태그 <jsp:include … />

- 내장 객체 pageContext의 메소드 include()와 같은 기능
  - <% pageContext.include("includesub.jsp"); %>
  - <jsp:include page="includesub.jsp" />

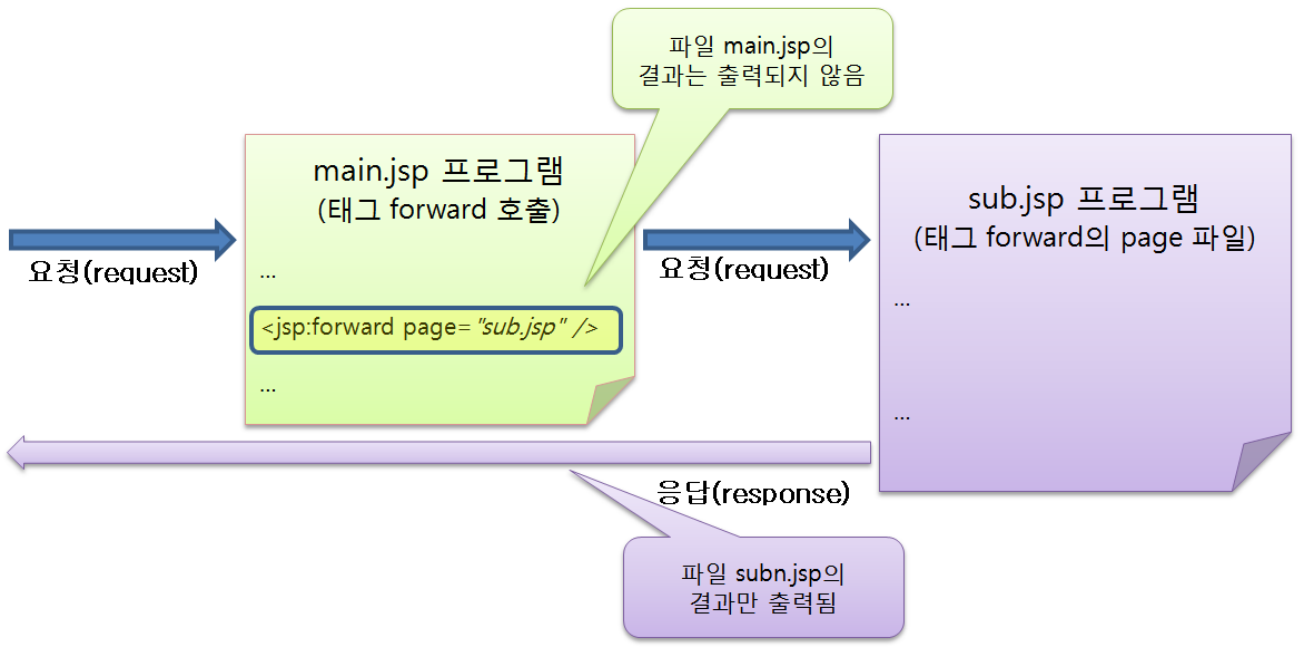
includedirective.jsp | includeaction.jsp | JSP 예제 : includeaction.jsp

```
1 <%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5 <title>JSP 예제 : includeaction.jsp</title>
6 </head>
7 <body>
8     <% int i = 12; %>
9     <% int n = 365; %>
10    1 년은 <%=i %>달 입니다.
11    <jsp:include page="includesub.jsp" />
12    1 년은 <%=n %>일 입니다.
13 </body>
14 </html>
```

액션 태그 forward

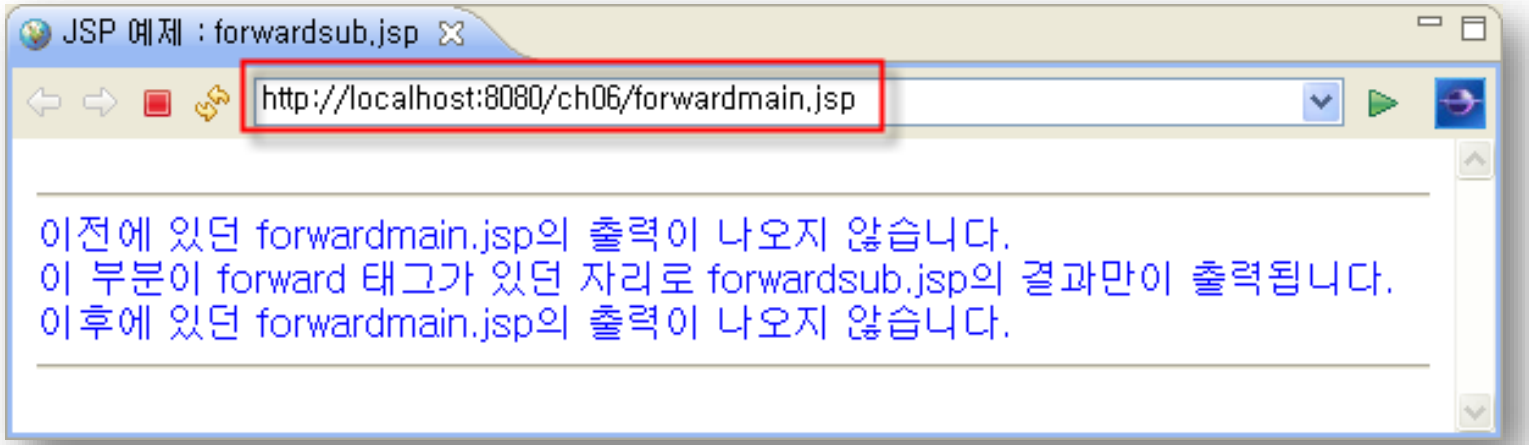
❖ 속성 page

- 속성 page에 지정한 JSP 페이지 또는 파일을 호출하는 기능
  - <jsp:forward page="*forwardsub.jsp*" />
- forward 태그가 있는 현재 페이지의 작업은 모두 중지 되고,
- 이전에 출력한 버퍼링 내용도 모두 사라지게 되어 출력이 되지 않으며
- 모든 제어가 page에 지정한 파일로 이동



태그 forward와 include의 차이

- 태그 include
  - page 속성에 지정된 페이지의 처리가 끝나면 다시 현재 페이지로 돌아와 처리를 진행
- 태그 forward
  - page 속성에 지정된 페이지로 제어가 넘어가면 다시 현재 페이지로 다시 돌아오지 않고 이동된 페이지에서 실행을 종료



pageContex.forward()

❖ 액션 태그 forward

- 실제 JSP 서블릿 소스에서 내장 객체 pageContext의 메소드 forward()로 대체
- pageContex.forward() 같은 기능을 수행
  - <% pageContext.forward ("send.jsp"); %>
  - <jsp:forward page="*send.jsp*" />

forward_jsp.java

```
42      try {
43        response.setContentType("text/html; charset=EUC-KR");
44        pageContext = _jspxFactory.getPageContext(this, request, resp
45                  null, true, 8192, true);
46        _jspx_page_context = pageContext;
47        application = pageContext.getServletContext();
48        config = pageContext.getServletConfig();
49        session = pageContext.getSession();
50        out = pageContext.getOut();
51        _jspx_out = out;
52
53        out.write("\r\n");
54        out.write("<html>\r\n");
55        out.write("<head>\r\n");
56        out.write("<meta http-equiv=\"Content-Type\" content=\"text/ht
57        out.write("<title>JSP 데젠 : forward.jsp</title>\r\n");
58        out.write("</head>\r\n");
59        out.write("<body>\r\n");
60        out.write("\t");
61
62      //pageContext.forward("send.jsp");
63
64        out.write('\r');
65        out.write('\n');
66        out.write('    ');
67        if (true) {
68          _jspx_page_context.forward("send.jsp");
69          return;
70        }
71        out.write("\r\n");
72        out.write("</body>\r\n");
73        out.write("</html>");
74      } catch (Throwable t) {
```

액션 태그 param

❖ 태그 param

- 태그 <jsp:include … >와 <jsp:forward … >와 함께 사용
- page에 지정된 페이지로 필요한 패라미터의 이름(name)과 값(value)을 전송하는 역할을 수행
  - 태그 param은 속성 name과 value를 제공

```
<jsp:include page="loginhandle.jsp" >
        <jsp:param name="userid" value="guest" />
        <jsp:param name="passwd" value="anonymous" />
</jsp:include>
```

태그 include에서 param 태그 이용

- 태그 include에서 지정한 인자의 전송

```
<jsp:include page="loginhandle.jsp" >
        <jsp:param name="userid" value="guest" />
        <jsp:param name="passwd" value="anonymous" />
</jsp:include>
```

```
<%
if (userid.equals("")) {
%>
        <jsp:include page="loginhandle.jsp" >
                <jsp:param name="userid" value="guest" />
                <jsp:param name="passwd" value="anonymous" />
        </jsp:include>
<%
} else {
%>
        <jsp:include page="loginhandle.jsp" />
<%
}
%>
```

태그 forward에서 param 태그 이용

- 태그 forward에서 지정한 인자의 전송

```
<jsp:forward page="forwardloginhandle.jsp" >
        <jsp:param name="snum" value="2010-3459" />
</jsp:forward>
```

```
<%
if ( userid == null && passwd == null ) {
%>
        <jsp:forward page="forwardloginhandle.jsp" />
<%
} else {
%>
        <jsp:forward page="forwardloginhandle.jsp" >
                <jsp:param name="snum" value="2010-3459" />
        </jsp:forward>
<%
}
%>
```

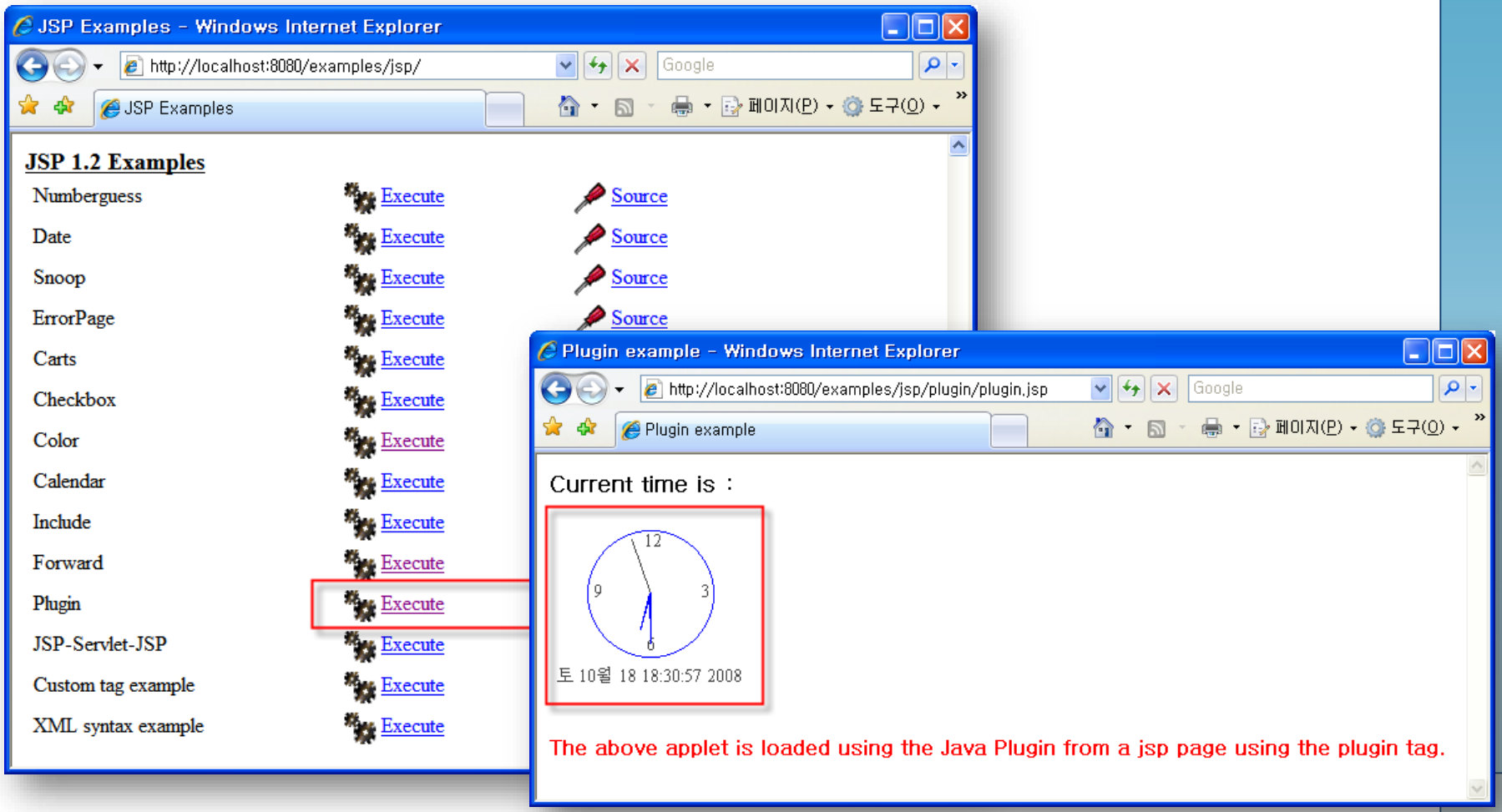
액션 태그 plugin

- plugin
  - 웹브라우져에서 자바 빈즈 또는 애플릿을 플러그인하여 실행하는 태그
  - 태그 plugin은 각기 다른 웹 브라우저에서 인식할 수 있도록 마이크로소프트 사의 IE 경우일 때는 OBJECT 태그로 만들어 주며, 넷스케이프 사의 경우, EMBED 형태의 태그로 만들어 줌

```
<jsp:plugin
    type = "bean | applet"
    code = "objectCode"
    codebase="objectCodebase"
    align="alignment“
    width = "width"
    nspluginurl = "url"
    iepluginurl = "url“ >
    <jsp:params name="paramName" value="paramValue" />
    <jsp:fallback> arbitrary_text </jsp:fallback> >
</jsp:plugin>
```

톰캣 플러그인 예제

❖ 플러그인 예제인 [plugin.jsp]



Thank You !

www.dongyang.ac.kr